CSCI 4131 – Internet Programming

Homework Assignment 5 - Introduction to Node.JS

Posted: October 27, 2018 **(Last Update: Friday, 11/2)**

DUE DATE: Friday November 9th at 2:00pm (afternoon)

Late Submission Date (with Penalty): Saturday, November 10th at 2:00am (Early Morning)

1 Description

The objective of this assignment is to introduce you web-server development with Node.js. We will provide most of the code that executes on the client-side (in your browser) and a bit of the server-side code. You are required to add/complete certain functions to complete the assignment. Node.js is basically JavaScript running a Web- server. Node.js uses an event-driven, non-blocking I/O model. So far, in this course we have used JavaScript for client-side scripting. For this assignment, we will use JavaScript for server-side scripting. Essentially, instead of writing the server code in Python, we will develop a basic web-server using JavaScript.

In this assignment, you will also use minimal amount of jQuery for AJAX and DOM manipulation. AJAX is used on client-side to create asynchronous web applications. It is an efficient means of requesting data from the server, capable of asynchronously receiving data from the server, and updating portions of a web page without reloading the entire web-page.

2 Preparation and Provided Files

I. The first step will be to get Node.js running on CSE lab machines. This can be accomplished as follows:

1. Log into a CSE lab machine.
2. Most of the CSE lab machines run version 8.9.4 of Node.js (as of 10/26/2018) which is the most current version. Vole runs version 8.11.4 which will also be sufficient for this assignment.
3. Open the terminal and type the following command to add the Node.js module:

```
module add soft/nodejs
```

4. The next step is to check the availability of Node.js. Type the following command to check the version of Node.js on the machine:

```
node -v
```

5. This will display the current installed version.

II. The second step is to create a Node.js project for this assignment. This can be accomplished as follows:

1. Open the terminal on a CSE lab machine.
2. Create a directory named <x500id_hw05> by typing the following command:

```
mkdir yourx500id_hw05
```

3. Go inside the directory by typing the following command:
   cd yourx500id_hw05

4. Having a file named *package.json* in Node.js project makes it easy to manage module dependencies and makes the build process easier. To create *package.json* file, type the following command:
   npm init

5. This will prompt you to enter the information. Use the following guideline to enter the information (The things that you need to enter are in bold. Some fields can be left blank.):

   name: (yourx500id_hw08) **yourx500id_hw05**

   version: (1.0.0) **<Leave blank>**

   description: **Assignment 5**

   entry point: (index.js) **<Leave blank> (**We will provide an index.js file for your use**)**

   test command: **<Leave blank>**

   git repository: **<Leave blank>**

   keywords: **<Leave blank>**

   author: **yourx500id**

   license: (ISC) **<Leave blank>**

6. After filling in the above information, you will be prompted to answer the question: "`Is this ok? (yes)`". Type **yes** and hit enter.
7. Now copy all the files present that are provided for this assignment to this directory: *yourx500id_hw05*
8. Change the permissions of all files and folders inside *yourx500id_hw05 directory to 777.*
9. Listing all the available files in the directory (using `ls -al`) should display the following:

```
-rwxrwxrwx  1 vaybhavshaw  staff  545 Oct 26 14:42 calendar.json
drwxrwxrwx  6 vaybhavshaw  staff  192 Oct 26 14:36 client
-rwxrwxrwx  1 vaybhavshaw  staff  715 Oct 26 15:11 createServer.js
-rwxrwxrwx  1 vaybhavshaw  staff  228 Oct 26 14:47 package.json
```

10. The project setup is now complete and we are ready to start the server.

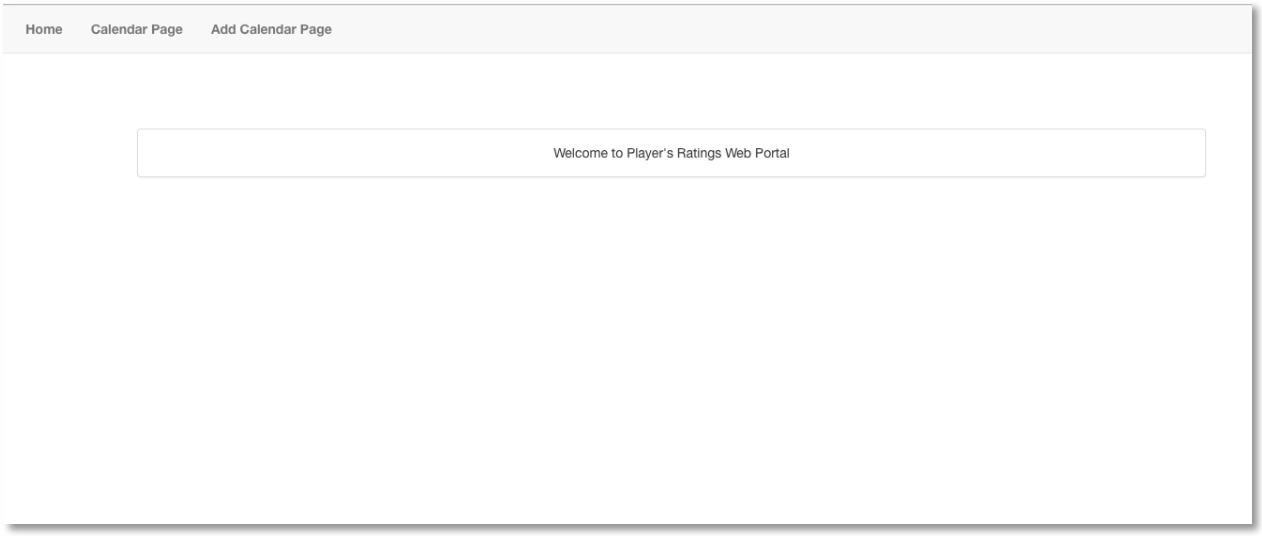
<u>To start the server, type the following command:</u>

node createServer.js

This starts the server and binds it to port 9000.

Now, using your browser on the same machine as the server (i.e., locally) open: **localhost:3000**

The following page should be displayed :



The following files are provided for this assignment:

You will only need to add code to the files highlighted in red.

1. **createServer.js**: This file contains the partially complete code for the node.js server.
2. **client/index.html**: Home page for this application.
3. **client/calendar.html**: Page which displays the list of calendar events. You need to fill in the TODO which would send a GET request to the Node.JS server to fetch calendar.json data and display it in a table
4. **client/addCalendar.html**: Form to add details about a new place. When the OnSubmit event is triggered, the form will POST the form data to the Node.JS server.
5. **calendar.json**: This file contains the list of calendar events in JSON format.

3 Functionality

Note: We advise you to complete the code changes for server before changing the code for client. All the server endpoints (APIs) can be tested using POSTMAN or curl.

Client

All the resources related to client have been provided in the client folder. The client folder has three HTML files (index.html, calendar.html, addCalendar.html).

You only need to change client/calendar.html file in the TODO section.

The file *calendar.html* has a table (id = "**calendarTable**") whose body is empty. You need to add code in *the TODO section* to dynamically populate the contents of the table after getting the

list of calendar details from the server. Add code in *calendar.html* file to implement the following functionality:

1. Using AJAX, query the *getCalendar* endpoint of the server using the GET method.
2. Upon successful completion of the AJAX call, the server will return the list of calendar entries (a JSON string).
3. Use the response returned to dynamically add and populate rows in the ' calendarTable' present in *calendar.html* page.
4. You can use of JavaScript or jQuery to achieve this.

Server

When the server starts, it listens for incoming connections on port 9000. This server is designed to handle only GET and POST requests.

GET requests:

1. The server has been designed to serve three different HTML pages to clients: *index.html, calendar.html,* and *addCalendar.html*
2. The server can also read and write to the list of calendar entries (in JSON format) by accessing the *calendar.json* file.
3. GET request for the index.html: The code for this has already been provided to you in the *createServer.js* file where the server is listening on the endpoint `"/"` and `"/index.html"`. **You do not need to add any code for this.**
4. GET request for the *calendar.html* page:
    a. When the **Calendar Page Tab** is clicked on the browser, a request is sent to the server to fetch the calendar.html file.
    b. You need to write code in createServer.js to listen for a request to the endpoint "./calendar.html" and return the file 'client/calendar.html' to the client
5. GET request from the *calendar.html* page:
    a. You need to add an endpoint that responds the GET request from the AJAX call in calendar.html page.
    b. You need to write code in createServer.js associated with the endpoint to fetch json data from calendar.json file and return the json data (a json string) to calendar.html (which will be used by calendar.html to dynamically build and populate the 'calendarTable').
6. GET request for the add*Calendar.html* page:
    a. When the **Add Calendar Tab** is clicked on the browser, a request is sent to the server to fetch the addCalendar.html file.
    b. You need to write code in createServer.js to listen to the endpoint "./addCalendar.html" and return the 'client/addCalendar.html' file to the client
7. GET request for any other resource: If the client requests any resource other than those listed above, the server should return a 404 error. The implementation is already provided in the code.

8. POST requests:
    - The server should process the form data posted by client. The form present in *addCalendar.html* allows user to enter details about a new calendar event and update the list of calendar events. The user enters the Event Name, Location, and the Date (Format: DAY TIME(24 Hr) eg: **THU 17:55**) in the form.
    - Details for a few events are pre-populated in the *calendar.json* file. Your job is to add the details of a new calendar event to this file and redirect the user to the *calendar.html* page after successful addition of the new course.
    - For this, you need add an endpoint respond to the "/postCalendarEntry" POST request from the addCalendar.html file. You need to write code associated with the "postCalendarEntry" endpoint to read the form data that is included in the POST request; add the new information to calendar.json file, and then redirect the user to calendar.html. The code for redirection is 302. Ensure that the newly added data does not change the format of the calendar.json file. **Hint**: You can use querystring module for parsing form data.

4 Submission Instructions

1. Zip your entire project and the name of the zipped folder should be your **x500id_nodejs**.

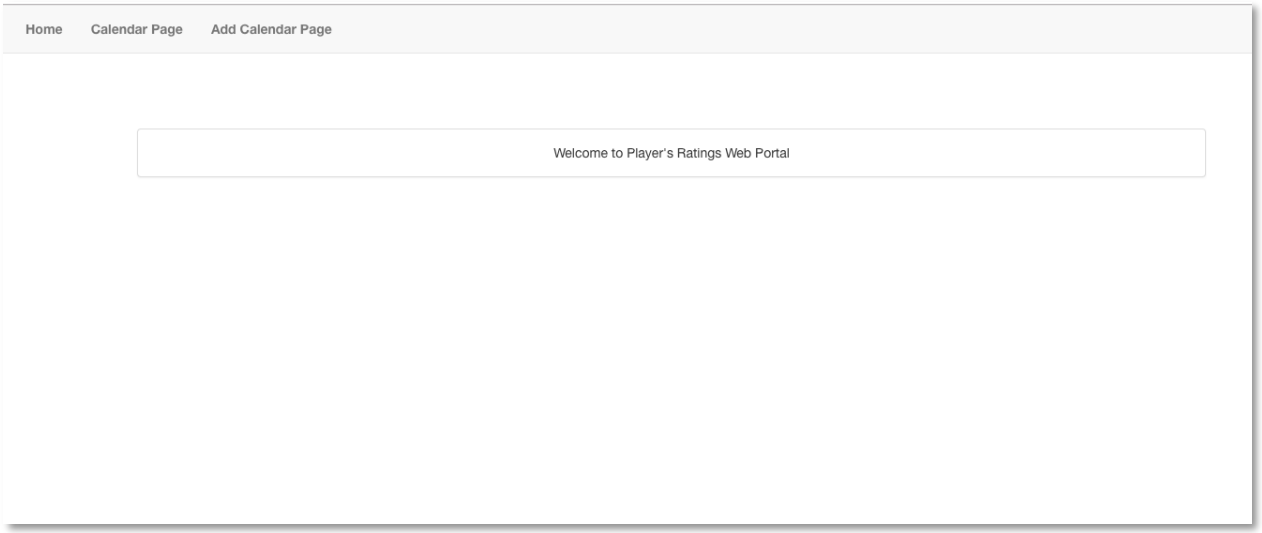
5 Evaluation

Your submission will be graded out of 100 points on the following items:

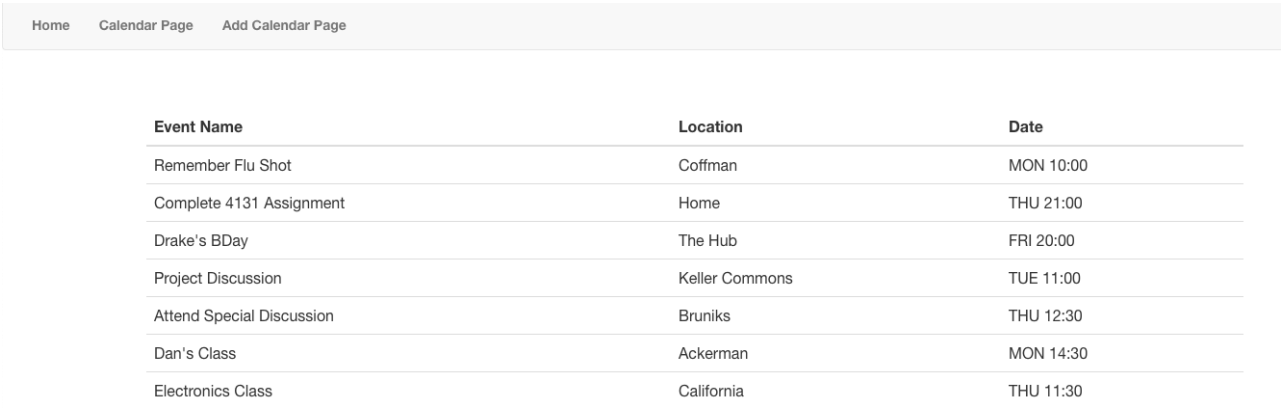
1. *calendar.html* is successfully returned by the server. (**15 points**).
2. *addCalendar.html* is successfully returned by the server (**15 points**).
3. Client successfully gets the list of calendar events from the server. The calendar events are dynamically added to the table present in the `calendar.html` page. (**30 points**)
4. *A form POST request to the sever's POST endpoint* successfully adds the details of the new calendar entry to `calendar.json` file (**30 points**).
5. The user is redirected to *calendar.html* page after successful addition of a new place (**10 points**).

6 Screenshots

index.html

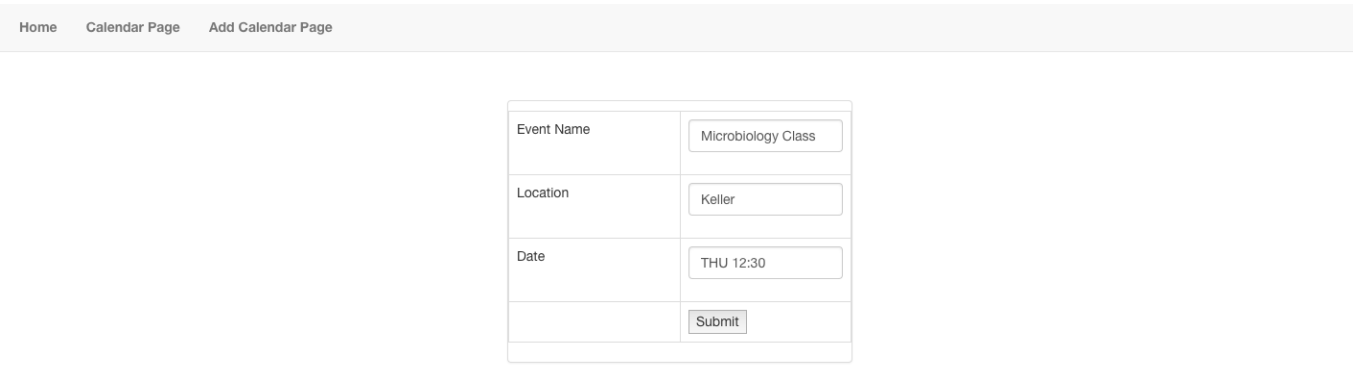


Initial display for calendar.html



Event Name	Location	Date
Remember Flu Shot	Coffman	MON 10:00
Complete 4131 Assignment	Home	THU 21:00
Drake's BDay	The Hub	FRI 20:00
Project Discussion	Keller Commons	TUE 11:00
Attend Special Discussion	Bruniks	THU 12:30
Dan's Class	Ackerman	MON 14:30
Electronics Class	California	THU 11:30

Form to add a new calendar event (displayed on the next page)


Home
Calendar Page
Add Calendar Page
Event Name
Microbiology Class
Location
Keller
Date
THU 12:30
Submit

calendar.html page displayed in the Browser after addition of a new calendar event